White paper
Intel® Xeon Processor-based servers
Intel® Xeon Phi™ Processor-based servers
Server-side Oracle Java* applications

Accelerating performance for server-side Java* applications

Intel® contributes significantly to both software and hardware optimizations for Java*. These optimizations can deliver performance advantages for Java applications that run using the optimized Java Virtual Machine (JVM), and which are powered by Intel® Xeon® processors and Intel® Xeon Phi™ processors. Developers do not need to recompile to get the benefit of these optimizations. The capabilities are already integrated into the latest Intel® processors and the latest Java Development Kit (JDK) release.

EXECUTIVE SUMMARY

Most people think that Intel develops and optimizes only hardware. In reality, Intel also invests heavily in software development to optimize software stacks for the latest, advanced servers. This effort is both in-house and in collaboration with runtime software communities and other application communities.

Intel's collaborations help developers benefit from powerful new capabilities designed into Intel® processor-based platforms. It also makes it easier for you to take advantage of both hardware and software optimizations, and maximize the performance of your applications. These software contributions help reduce the burden typically required to shift legacy applications to more powerful, advanced platforms. Since 2006, these team efforts have delivered a performance improvement of more than 110x for Java* applications that run on servers powered by Intel® Xeon® processors.[1,2]

This paper describes:

- Key architectural advancements that benefit Java applications which run on the latest Intel Xeon processors and Intel® Xeon Phi™ processors.
- Some of Intel's software contributions in team efforts with Oracle* and the OpenJDK* community. The collaboration work includes enabling platform capabilities, improving performance, optimizing microarchitecture for Java applications, improved libraries, and tuning Java virtual machines (JVMs) for specific Java JVM frameworks and applications, such as Apache Hadoop* and Apache Spark*.
- How both hardware and software optimizations can benefit your Java applications.
- Techniques and strategies you can use to take advantage of these features.

It is important to note that you do not have to recompile your Java application to see these new benefits. By running your applications on the latest Intel processor-based platform with the latest Java release, your application can automatically gain the maximum benefit from both hardware and software optimizations.

Intel®, Oracle*, and the Java* community collaborate to optimize the Java software stack

To maximize Java performance, software engineers from both Intel® and the Java* community work closely to optimize Java.

As Intel releases new platform features, Intel and Oracle* collaborate to identify specific areas where the JVM can take advantage of Intel's cutting-edge architecture.

Intel and the Java community then work together to identify and implement optimizations for these platform capabilities.

This helps the JVM deliver the highest performance possible across a broad range of applications, including enterprise, Web, cloud, and Big Data applications.

INTEL® AND ORACLE*: BUILDING A BETTER JAVA* FOUNDATION

Oracle Java* is one of the most deployed software platforms today. According to Oracle, it is now used in over three billion devices. Today, Java powers a massive array of usages from cloud, Big Data, enterprise, and financial services applications, to games and personal productivity software.

When Java first began to impact the market, Intel immediately began collaborating with key Java virtual machine (JVM) industry vendors on optimizations. The intention was to optimize JVM performance on Intel processor-based servers, so that Java applications would see maximum performance. Since 2006, these team efforts have delivered a performance improvement of more than 110x for Java applications that run on servers powered by Intel® Xeon® processors.[1,2]

Over the course of 20 years, this history of software collaboration includes:

- **BEA JRockit* JVM.** Intel began work with BEA in 2002, and continued this collaboration after Oracle acquired BEA in 2008.
- **Sun Microsystems Hotspot* JVM.** Intel began work with Sun Microsystems in 2007, and continued this team effort after Oracle acquired Sun in 2009.
- **Oracle.** Intel signed an Oracle Contributor Agreement in 2014, and began contributing to OpenJDK* in 2015.
- **Oracle Hotspot* and Oracle JRockit*.** Intel has continued a close collaboration with Oracle on the combined Hotspot and JRockit teams.
- **OpenJDK and JVM.** Since 2015, Intel software engineers have worked not just with Oracle, but directly with the open source OpenJDK community and Oracle JVM engineers. This effort is focused on identifying and providing specific optimizations that take advantage of the latest Intel® Architecture (IA) enhancements.

Because of Intel's ongoing collaboration with the Java community, organizations can realize significant performance gains using the latest Java release running on servers powered by the latest Intel processors.

New Intel® processors deliver advanced capabilities for Java

Most developers know that, with each new generation of processors, Intel delivers powerful new technologies for new, advanced platforms. What you might not yet know is that Intel has been designing optimizations for Java applications directly into the silicon.

As with Intel's work in enhancing the JVM, these hardware optimizations are developed in collaboration with Oracle and the OpenJDK community. The work supports and enables new capabilities that include: better memory and I/O performance, larger on-die cache, non-inclusive cache, non-uniform memory access, cluster nodes, and many other high performance capabilities.

For server-side Java developers, two new Intel® processor families are particularly important:

- Intel® Xeon® Scalable processor family
- Intel® Xeon Phi™ processor family

Intel® Xeon® Scalable processor

The new Intel Xeon Scalable processor provides the foundation for a powerful data center platform. This feature-rich processor family is highly versatile, and creates an evolutionary leap in both agility and scalability for data centers. It is a new level of performance for platform convergence and capabilities across compute, storage, memory, network, and security applications. Enterprises and cloud and communications service providers can now drive forward with their most ambitious digital initiatives and see the performance needed to deliver the services they want to offer.

Intel® Xeon Phi™ processor

The Intel Xeon Phi processor is a bootable host processor that delivers massive parallelism and vectorization. It is designed to support the most demanding high performance computing (HPC) applications. The Intel Xeon Phi processor integrates several Intel "firsts:"

- First processor that delivers the performance of an accelerator with the benefits of a standard host CPU
- First processor to integrate high bandwidth memory and fabric technology

The Intel Xeon Phi delivers the demanding performance needed for highly parallel workloads, high performance modeling and simulation, visualization, and data analytics.

Intel and Oracle continue to enhance the Java software stack

Intel's collaborations with the Java community to optimize the Java software stack includes support and enabling of many aspects of application development. For example, Intel's collaborations include work to enable platform capabilities in the JVM, improve JVM performance, optimize Intel Architecture for Java applications, provide improved libraries as part of the Java Development Kit (JDK), and tune JVMs for specific Java frameworks and applications.

Optimized Java sees powerful gains when running on new Intel® Architecture

Intel's collaboration work makes it easier for you to take advantage of cutting-edge Intel Architecture just by using the latest Java release. For example, Intel, Oracle, and the Java community collaborated to develop many new JDK 9 enhancements. These include: extended just-in-time (JIT) auto-vectorization, optimized java.lang.Math methods to improve HPC, accelerated encryption, and other demanding applications.

In addition, Intel has worked closely with Oracle and OpenJDK to optimize Java for today's demanding applications and frameworks:

- **Big Data applications.** For Big Data, Intel collaborated with the Java community to optimize the JVM for analytics, predictive modeling, genomic research, and other Big Data applications. This teamwork includes optimizations for Apache Hadoop, Apache Spark, Apache HBase,* and Apache Cassandra.*
- **Machine learning frameworks.** Intel is also working with the Java community to optimize the JVM for machine learning frameworks. These frameworks include Apache SparkML* and Apache H20*.

The new JVM 9 optimizations can make it easier for you to port and develop applications that take advantage of new hardware capabilities and the performance they can deliver.

No need to recompile

It is important to note that you do not have to recompile your Java application to see the benefits of the hardware and software optimizations. Intel has integrated the hardware optimizations directly into the latest Intel Xeon processors. By running your applications on the latest Intel processor-based platform with the latest JDK release, you can automatically gain the benefits of Java-optimized hardware and the optimized software stack.

"Intel® and Oracle® have been partnering closely with the OpenJDK community to deliver the highest performance with a lower footprint and scalable heap for the upcoming JDK 9 on the latest Intel platforms as the Intel(r) Xeon(r) Platinum 8100 processor. JDK 9 overall optimizations will significantly enhance Big Data, Cloud, Microservices, HPC and FSI applications with enhanced vectorization support, AVX-512 code generation and intrinsic methods, optimized math libraries, cryptography, checksum, compression acceleration and compact strings."*

— Bernard Traversat
Vice President of Development
Java SE Platform
Oracle

Older versions of JDK can still see increased benefits

To get the best performance from your Java applications, you should run the latest version of Java on the latest Intel processor-based platform. However, even older Java versions typically perform better simply by running on newer Intel platforms.

Performance gains: Intel and Oracle continue to enhance Java performance

Since 2006, the team efforts of Intel and the Java community have delivered a performance improvement of more than 110x for Java applications that run on servers powered by Intel Xeon processors.[1,2]

For example:

- **Performance gain: 32.34x.**[1] From 2006 to 2013, the collaboration of Intel and the Java community produced benchmark scores with significant performance gains of over 30x, as shown in Figure 1.[1]
- **Performance gain: 3.42x.**[2] From 2015 to 2017, the collaboration of Intel and the Java community developed optimizations that produced benchmark scores with performance gains of 3.4x, as shown in Figure 2.[2]
- **Performance gain: 3.96x.**[3] In the last four years, five generations of Intel Xeon processors have produced benchmark scores with a combined 4x performance improvement for enterprise applications alone.[3]

The next several graphs show performance data for various Intel processor generations and Java releases.

PROCESSOR FEATURES THAT BENEFIT JAVA PERFORMANCE

Intel continues to drive the evolution of the Intel Xeon processor E7, E5, E3 families, as well as the new Intel Xeon Scalable processor and Intel Xeon Phi families. These processors now deliver even more cores, higher performance, larger cache, improved energy efficiency, and other new, advanced capabilities. Because these capabilities are built into the silicon, your server-side Java applications can see significant performance gains simply by running on the latest Intel processor-based platform.

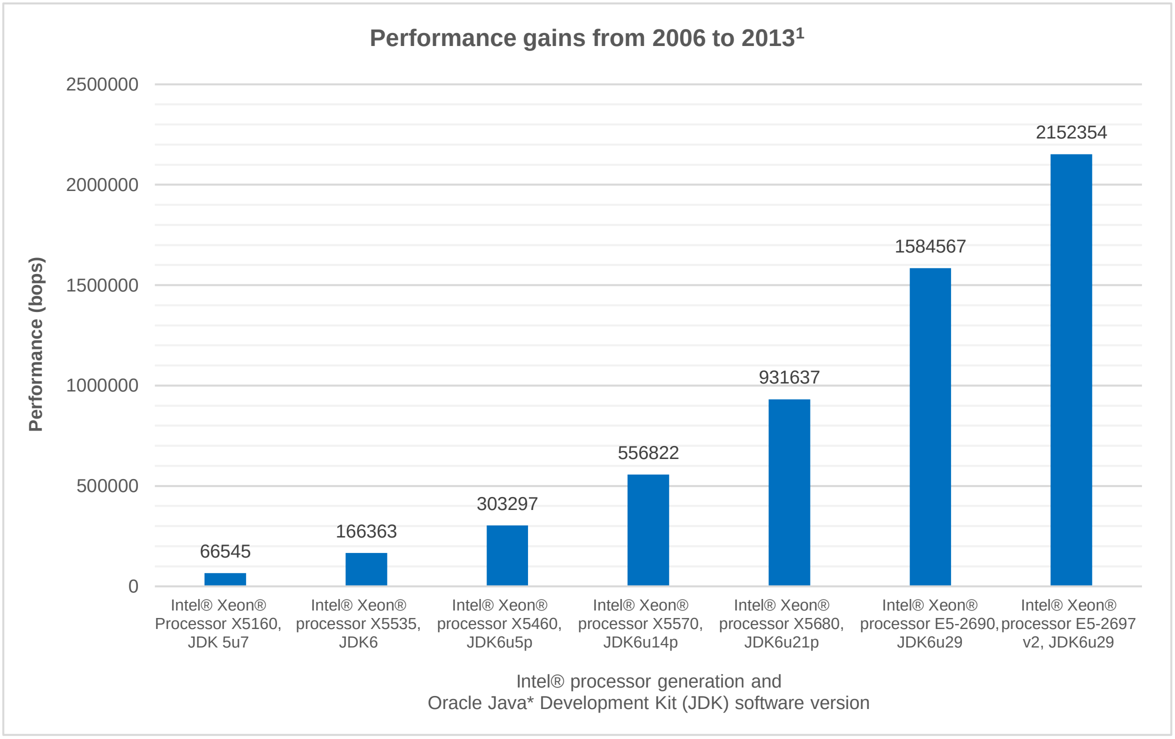
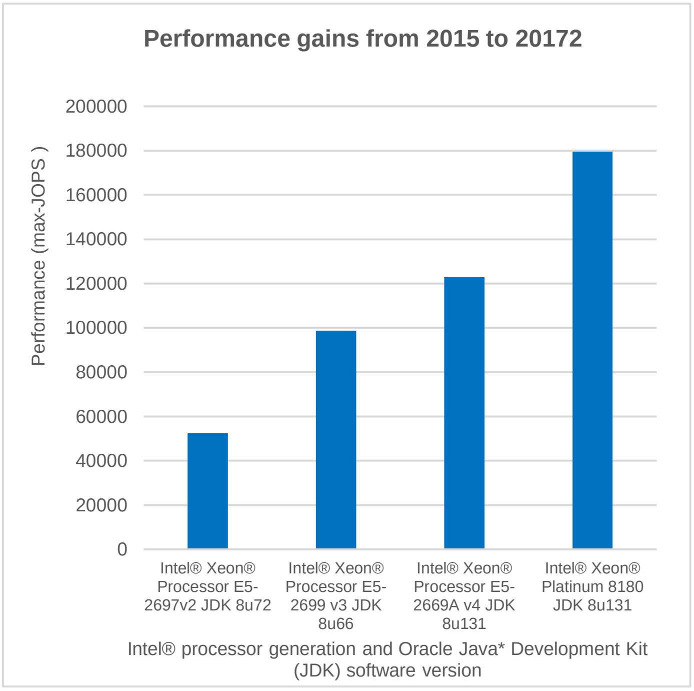


***Figure 1. Performance gains from 2006 through 2013**, across 7 generations of combined Intel® processors and Java* Development Kit software. The Standard Performance Evaluation Corporation SPECjbb2005* benchmark used for these tests measures business operations per second (SPECjbb2005 bops).*[1]



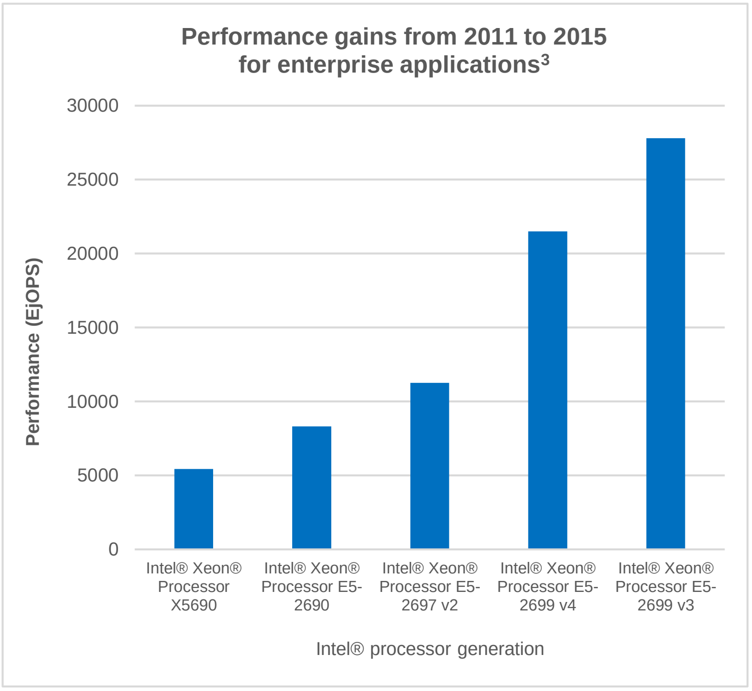


Figure 2. Performance gains from 2015 through 2017, *across 4 generations of combined Intel® hardware and Java* Development Kit software. The Standard Performance Evaluation Corporation SPECjbb2015* benchmark used for these tests measures performance by maximum operations per second (SPECjbb2015-MultiJVM max-jOPS).*[2]

Figure 3. Performance gains from 2011 to 2015, *across 5 generations of Intel® processors and Oracle WebLogic Server* software. The Standard Performance Evaluation Corporation SPECjEnterprise2010* benchmark used for these tests uses enterprise jAppServer operations per second (EjOPS) as the performance metric.*[3]

This discussion describes just some of the capabilities from which your Java applications can benefit.

New memory technologies

Better memory and I/O performance

Intel Xeon processors — including the new Intel Xeon Scalable processor family and the Intel Xeon Phi family — deliver increased memory and I/O performance for Java applications.

Intel Xeon processors include Intel® QuickPath Interconnect Technology (Intel® QPI) and integrated memory controllers. These technologies deliver faster core-to-core and core-to-memory communications.

Today's Intel Xeon processors also include larger shared, on-die cache than previous generations. That keeps more data closer to the processor cores for fast access. The shared cache architecture, along with improved cache coherency algorithms, helps to reduce cache misses.

The new Intel Xeon Scalable processor family supports non-inclusive cache and Intel® Ultra Path Interconnect (Intel® UPI). Both the Intel Xeon Scalable processor family and the Intel Xeon Phi family x200 support up to 6 channels of memory per socket. This increases the memory bandwidth and capacity available to applications. To realize maximum throughput of that capacity, these processor families also include a new high bandwidth mesh interconnect architecture to connect all cores and memory.

Powerful NUMA and MCDRAM capabilities for Java* applications

NUMA (non-uniform memory access) is a shared memory architecture, while multi-channel DRAM (MCDRAM) provides integrated, on-package, high bandwidth memory.

The combination of NUMA and MCDRAM can significantly improve data flow to and from the processor cores. This helps the processing cores spend less time waiting for data and more time actively processing it. NUMA and high bandwidth MCDRAM are particularly important for delivering breakthrough performance for memory-bound Java workloads.

The two memory technologies also help accelerate garbage collection (GC) processes. Such processes are used to free up memory space in Java applications by eliminating software objects that are no longer needed. Many Java applications are GC-intensive, so speeding up these processes can substantially improve overall performance.

Non-uniform memory access (NUMA) and cluster modes

Lower latency and higher local memory bandwidth

NUMA, or non-uniform memory access, is a shared memory architecture. In this architecture, each processor that makes a set of CPUs has a local memory and a last-level cache (LLC) slice associated with it. Any processor in the system can access all memory on the system, although remote access is slower.

- **NUMA.** Intel Xeon processors from the Intel Xeon Scalable processor 8180 onwards support NUMA.
- **Two NUMA nodes.** Intel Xeon Processor Families E5/E7 v3 and v4 support cluster-on-die capabilities, where each processor supports two NUMA nodes.
- **Sub-NUMA clustering.** Intel Xeon Phi family x200 and Intel Xeon Scalable processor families support sub-NUMA clustering.

The sub-NUMA clustering and cluster-on-die modes have the lowest local memory latencies and highest local memory bandwidth. In these modes, each cluster is exposed as a separate NUMA domain to the operating system. NUMA-optimized software will gain the most benefit from these memory access and clustering capabilities.

To benefit from NUMA, a Java application made of multiple JVMs can bind a JVM to a NUMA node. For Java applications that run across nodes, you can see additional performance benefits by enabling the NUMA optimization options of the JVM.

Higher bandwidth memory

Improves data flow and garbage collection

The Intel Xeon Phi processor family x200 includes up to 16 GB of integrated, on-package, high bandwidth multi-channel DRAM (MCDRAM) memory. This provides up to 490 GB/s of sustained memory bandwidth.

You can configure MCDRAM in any of these ways:

- Third-level cache (cache mode)
- Distinct NUMA node (flat memory)
- Part cache and part memory (hybrid mode).

MCDRAM works with NUMA to improve data flow to and from the processor cores, and accelerate garbage collection.

Intel® Hyper-Threading Technology[4]

Increases processor utilization and throughput

Newer Intel processor cores include Intel® Hyper-Threading Technology (Intel® HT Technology). These processor cores can simultaneously execute multiple hardware threads.[4] While one thread is waiting for data or instructions, the other can remain active to further increase processor utilization and overall throughput. This allows the processor to simultaneously run demanding applications and still maintain system responsiveness.

Intel Xeon processors can simultaneously process multiple hardware threads to improve performance for Java applications:

- Intel Xeon family processor cores: 2 hardware threads.
- Intel® Xeon® Scalable processor cores: 2 hardware threads
- Intel Xeon Phi x200 family processor cores: 4 hardware threads.

Except for a few niche workloads, Intel software engineers have observed that Intel HT Technology typically improves performance by 5 percent to 35 percent for threaded Java applications.[5]

Intel HT Technology also helps servers support more users per server even in peak loads. At the same time, the technology provides balanced performance across the consolidated workloads. Intel HT Technology helps keep your systems efficient, while providing headroom for future business growth and capabilities for new Java solutions.

Intel® Turbo Boost Technology

Boosts performance of cores at peak workloads

With Intel® Turbo Boost Technology, Intel Xeon processors and Intel Xeon Phi processors can automatically operate cores above their rated frequency during heavy workloads.[6] This enables the processor to respond intelligently to demands and deliver maximum performance when needed.

Intel Turbo Boost Technology automatically and dynamically applies a frequency boost to each core, so performance can be tailored to match specific workloads. In particular, Intel Turbo Boost Technology can automatically improve overall throughput for demanding Java applications. It can also automatically improve response times for time-sensitive Java applications.

If you're working on newer Intel hardware, you do not need to incorporate additional optimizations in order to take advantage of Intel Turbo Boost Technology. The technology and its Java-oriented optimizations are already fully integrated into newer Intel® processors.

Intel® Intelligent Power Technology

Automatically performs fine-grained, policy-based energy efficiency

As with Intel Turbo Boost, Intel® Intelligent Power Technology (Intel® IPT) is designed directly into Intel Xeon processors. This technology enables automatic, fine-grained, policy-based control of server energy states. With this powerful technology the processor can automatically optimize performance and power consumption across all workloads.

Intel IPT allows the operating system to put both processor power and memory into the lowest available energy states needed to support current workloads without compromising performance. A processor with this technology can independently manage power consumption for individual cores. This includes putting one or more cores into low-power idle states as needed. With Intel IPT, you can reduce power consumption, while also reducing server idle power by up to 50 percent.[7]

Most Java applications have a range of power utilization patterns in production environments. The Intel IPT capabilities can be configured in BIOS, so you can choose the best balance of power versus performance for your specific application. When the processor automatically and dynamically tailors power consumption to the workload, you can often see significant power savings, with little or no impact on performance or response times.

Accelerated floating-point operations

Speeds up execution of math functions

Java is used in a large number of math-intensive applications, including the management and analysis of Big Data. But, with the increasing width of Single Instruction Multiple Data (SIMD) and the richness of instructions not directly accessible to the Java programmer, Java has seen limited use where advanced floating-point intensive operations are common. This includes areas such as high-performance computing and media processing. These areas can now benefit from the JVM's support for Intel® Advanced Vector Extensions (Intel® AVX and Intel® AVX2).

Accelerated floating-point operations speed up math-intensive applications

Intel® Advanced Vector Extensions 512 (Intel® AVX-512) was introduced into the latest Intel processors. This technology improves the performance of floating-point operations, accelerate character encoding, and provides other benefits for Java* applications. Intel AVX and Intel AVX2 are built into Intel processors:

- **Intel® Xeon Phi™ processor.** Intel® AVX-512 was originally introduced with the Intel Xeon Phi processor family x200 product line. There are certain Intel AVX-512 instruction groups (AVX512F AVX512CD) that extend the vector width to 512 and are common to the Intel Xeon Phi processor product line, as well as the new Intel Xeon processor Scalable family.
- **Intel® Xeon® Scalable processor family.** The new Intel Xeon Scalable processor family introduces new Intel AVX-512 instruction groups (AVX512BW and AVX512DQ) as well as a new capability (AVX512VL) to expand the benefits of the technology. The AVX512DQ instruction group is focused on new additions for benefiting HPC workloads. This includes workloads such as oil and gas, seismic modeling, financial services industry, molecular dynamics, ray tracing, double-precision matrix multiplication, fast Fourier transform and convolutions, and RSA cryptography.

Intel AVX expands Intel® Streaming SIMD Extensions (SSE) instructions. For example, with Intel AVX, wider vectors improve floating-point operation performance. Intel AVX2 provides wider vectors for integer arithmetic operations.

Optimizations for Intel AVX/AVX2 also further accelerate character encoding, string and array operations, and other mathematical functions in Java. For example, one optimization uses Intel AVX/AVX2 to increase character encoding to 16 or 32 characters at a time, instead of character by character. This improves the performance of applications that use international string encoding.

For example, the Strings optimization allows JIT compiled code to execute up to twice the operations per iteration as un-optimized Java applications.

Accelerated encryption is integrated into Intel® processors

Intel® Advanced Encryption Standard-New Instructions (Intel® AES-NI) was first introduced in the Intel® Xeon® processor x5680. Intel AES-NI is available in all newer Intel® processors:

- Intel® Xeon® E5/E5v2/E5v3/E5v4
- Intel® Xeon® Scalable processor
- Intel® Xeon Phi™ processor family x200

Both Java Development Kit* 9 (JDK 9) and JDK 8 have been optimized for accelerated encryption via Intel AES-NI.

New Intel bit-manipulation instructions

Improves performance of bit manipulation sequences

Intel advanced bit manipulation instructions (BMI1) have been introduced in the Intel Xeon E3/E5/E7 v3 family. These instructions are useful for compressed database, hashing, large number arithmetic, cryptography, and a variety of general purpose codes.

The BMI1 optimization for the JVM allows the Just-in-time (JIT) compiler from JDK 8 update 20 onwards to recognize the Java pattern for these instructions. This optimization then automatically generates intrinsic functions for those instructions when possible. For Java applications, the BMI1 optimization improves the overall performance of bit manipulation code sequences.

Intel floating-point multiply accumulate

Increase peak flops and improve precision of transcendental math

Floating-point multiply accumulate (FMA) instructions have been supported in Intel architecture since the Xeon E3/E5/E7 v3 family processors. These instructions significantly increase peak floating point operations (flops). They also provide improved precision that increases the performance of transcendental mathematics.

The instructions are broadly usable in high performance computing, professional quality imaging, and face detection.

Before the introduction of JDK 9, Java users could not benefit from FMA instructions. With JDK 9 comes the integration of Math.fma() as part of Java runtime environment, or JRE. The JVM now generates underlying FMA instructions automatically, via a new API. This FMA optimization can increase the performance of HPC, artificial intelligence, and machine-learning applications.

Accelerated encryption

Enables wider use of encryption and decryption without a performance penalty

Banking, healthcare, and other institutions often deploy Java applications that protect sensitive data during transit. However, software-based encryption is compute-intensive. Because of this, some organizations have not implemented secure, in-house data transit. These businesses often assume they cannot afford the performance degradation typically seen with such security applications and methodologies.

Intel has taken a two-pronged approach to solve that problem. First, Intel introduced Intel® Advanced Encryption Standard-New Instructions (Intel® AES-NI) in the Intel® Xeon® processor x5680. The instructions are now built into all newer Intel Xeon processors. This enables on-chip encryption and decryption in these processors, and accelerates these operations as much as 4x.[8]

Second, Intel worked closely with the Java community to optimize the JVM for encryption and decryption. This work included optimizing the Java cryptography architecture (JCA) to define a provider framework. A default provider is defined, as well as several cryptography stubs. Along with the hardware-enhanced Intel AES-NI, these optimizations enable a much wider use of data encryption and decryption without a performance penalty.

By taking advantage of Intel AES-NI, Java applications can now benefit from much faster encryption and decryption. In turn, this will help motivate more businesses to secure data more fully in-house, not just off-premises.

PCLMULQDQ

Improves network and file system performance

With the explosion of high-speed networking, 10 gigabit/second Ethernet (10GbE) networks are becoming commonplace. Along with that growth has come a dramatic increase in storage demands over the past 10 years.

One of the challenges this created for Java developers is dealing with the generation of residue from cyclic redundancy checks (CRC). For example, in file systems like the Apache Hadoop Distributed File System (HDFS)*, the processor must generate a CRC32 checksum at network speeds for every 512 bytes of data.

To help resolve this problem, Intel integrated a new PCLMULQDQ instruction into Intel processors. The PCLMULQDQ instruction performs carry-less multiplication of two 64-bit operands. In Intel processors, the instruction accelerates CRC computations for HDFR and similar file systems. This increases file checksum performance, as well as compression/decompression checksum performance. It then speeds up management of residue from CRCs. With these improvements, the new instruction increases the performance of both network and file systems overall.

Intel continues to improve the performance of PCLMULQDQ with every generation of processors. Intel also works with the Java community to enable both JVM 8 and JVM 9 to use this instruction. Depending on how often your application uses CRCs, taking advantage of the PCLMULQDQ instruction in Java applications can significantly improve Java performance for Big Data and other environments.

For example, Intel collaborated with the Java community to develop a new CRC generation algorithm for JVM 8. More recently, Intel worked with the Java community to implement a new optimization to support CRC32C in JVM 9. (JVM 8 uses CRC32.) This optimization is a new CRC32C method that further increases the performances of checksum calculations for big data and other applications.

Intel® Transactional Synchronization Extensions

Enables fine-grain lock performance with coarse-grain locks

Intel® Transactional Synchronization Extensions (Intel® TSX) is made of restricted transactional memory (RTM). This type of memory supports a set of instructions you can use to specify a critical section of code. The RTM hardware then executes the specified critical sections (also referred to as transactional regions) transactionally. For Java applications, the TSX optimization provides the benefit of fine-grain lock performance with coarse-grain locks.

Here's how Intel TSX works, depending on the success or failure of the transactional execution:

- **Transactional execution completes successfully.** The hardware ensures that all memory operations performed within the transactional region will appear to have occurred instantaneously, when viewed from other logical processors. This process is called an atomic commit.
- **Transactional execution is unsuccessful.** The processor cannot commit the updates atomically. In this case, the processor aborts the transaction and discards all updates performed in the region. The processor then restores the architectural state so that it appears as if the optimistic execution never occurred. Finally, the processor resumes execution non-transactionally at the user-specified abort handler. Depending on the software abort handler in place, the transaction may be retried or the lock may be explicitly acquired to ensure forward progress.

Intel TSX is supported on Intel Xeon platform E7 v3 and E5 v4 onwards. The JVM supports TSX from JDK 8 update 20 onwards. You can enable TSX for your Java applications using JVM command line options.

Intel® QuickAssist Technology

Accelerates and compresses cryptographic workloads

Intel® QuickAssist Technology (Intel® QAT) accelerates and compresses cryptographic workloads. Intel QAT does this by offloading data to hardware that is capable of optimizing cryptographic functions. With Intel QAT, you can more easily integrate built-in cryptographic accelerators into your network, storage, and security applications.

In the case of the Intel Xeon processor Scalable family, Intel QAT is integrated into the hardware of the Intel® C620 series chipset. Here, Intel QAT offers outstanding capabilities that include 100 Gbs encryption, 100 Gbs compression, 100 kops RSA, and 2k decryption.

The optimization that accelerating and compressing cryptographic Java-based workloads enables wide use of encrypted data in transit.

Intel® Omni-Path Architecture

Delivers scalable performance for HPC workloads

Intel® Omni-Path Architecture (Intel® OPA) is an element of Intel® Scalable System Framework. It delivers the performance for tomorrow's HPC workloads, with the ability to scale to tens of thousands of nodes—and eventually more. The Intel OPA 100 Series product line is an end-to-end solution of PCIe* adapters, silicon, switches, cables, and management software.

As the successor to Intel® True Scale Fabric, this optimized HPC fabric is built upon a combination of enhanced intellectual property and Intel® technology. Both the Intel Xeon processor Scalable family and the Intel Xeon Phi processor family x200 support Intel OPA in one of these forms:

- Intel® Omni-Path Host Fabric Interface 100 Series add-in card
- Specific processor model line (SKL-F, KNL-F) within the processor family that has a host fabric interface integrated into the processor

The architecture is able to provide up to 100 Gb/s per processor socket.

OPTIMIZING THE JVM TO MAXIMIZE PERFORMANCE GAINS

Every major transition in processor architecture and platform capability requires that JVM developers rethink their performance and optimization techniques. This lets you maximize the performance gains delivered by the new generation of servers.

Enhancements in Intel Xeon processors and Intel® Xeon Phi processors have created many opportunities for Intel software engineers and their counterparts at leading JVM companies to tune and optimize code. These optimizations can help shorten your own development time and make your Java code more efficient.

Tables 1, 2, and 3 (next several pages) summarize some of the key optimizations developed by Intel and the Java community to boost the performance of your Java applications.

Table 1. Java* development kit (JDK) 9 optimizations that help boost Java performance

JDK 9 optimizations	Description	Performance benefit
Intel AVX512	Just-in-time (JIT) compiler auto-vectorization is extended to 512 bit in order to generate AVX-512 code. Tail processing is also improved and the reduction operation is supported in a vector loop. The array and string intrinsic methods are enhanced to use AVX-512 bit instructions.	When the Java* virtual machine (JVM) is invoked with the -XX:+UseAVX=3 option, this optimization improves the performance of floating-point and integer operations, as well as operations that contain array and string manipulation.
Math Libraries	Optimized trigonometric and transcendental java.lang.Math methods include: exp, pow, log, log10, sin, cos and tan.	Improves the performance of applications for Big Data, machine learning, financial option pricing, and high performance computing (HPC).
Compact Strings	The compact strings optimization uses byte array instead of char array to store characters with an encoding byte field. It changes the internal representation of String class, while preserving compatibility. The characters of String are encoded as either UTF-16 (two bytes per character) or ISO-8859-1/Latin-1 (one byte per character).	Improves space efficiency of the String class and related classes. String methods are highly optimized using Single Instruction Multiple Data (SIMD) instructions where possible. This optimization allows JIT compiled code to perform up to twice the operations per iteration as un-optimized Java applications.
Cryptography acceleration	Java cryptography architecture (JCA) defines a provider framework, with Oracle SunJCE* as the default provider. This cryptography acceleration optimization provides cryptography stubs for the SunJCE provider, using AES-NI, SHA-NI, PCLMULQDQ, ADCX/ADOX and AVX/AVX2; in order to accelerate AES-CTR, AES-CBC, AES-GCM, SHA1, SHA256 and SHA512.	Increases data encryption performance for storage and network transactions. This enables wide use of encrypted data in transit.
Compression acceleration	JDK 9 uses system Zlib instead of a statically linked (bundled) Zlib. This allows applications to override the system Zlib with a hardware/software accelerated library through LD_PRELOAD and LD_LIBRARY_PATH environment variables*.	Provides a compression acceleration feature for hardware and software, for applications that use compression extensively. Applications that typically use extensive compression include genomics, big data, and enterprise applications.
CRC32C support	This is a new crc32c method in java.util.zip for iSCSI CRC, as defined in RFC 3720, with the CRC polynomial as 0x1EDC6F41, and an optimized implementation using the Intel CRC32C instruction.	Increases the performances of checksum calculations for big data and other applications, through the use of the new API.
Lexicographic array compare	These are new compare and compareUnsigned methods in java.util.Arrays, for which the JIT compiler generates efficient code, using Intel SSE/AVX/AVX2/AVX- 512 instructions.	Increases the performance of big data applications that use arrays as keys.
Floating point multiply Accumulate (FMA)	New method in java.lang.Math for floating point multiply accumulate. The JIT compiler uses this to generate the FMA instruction.	Increases the performance of HPC, artificial intelligence, and machine-learning applications; through use of the new API.
Socket Reuse	SO_REUSEPORT removes the traditional 1:1 assignment between listen socket and IP:PORT pair; and enables multiple sockets listening to the same address and port.	Improves the scalability and parallelism of network traffic handling. This provides better throughput and lower latency through the use of the new SO_REUSEPORT option in java.net.StandardSocketOptions.

Table 2. Java* development kit (JDK) 8 optimizations that help boost Java performance

JDK 8 optimizations	Description	Performance benefit
Intel® Advanced Encryption Standard-New Instructions (Intel® AES-NI)	This is hardware-based acceleration of encryption and decryption via new Java* virtual machine (JVM) intrinsic functions for AES-CTR and AES-CBC using Intel AES-NI.	Increases the performance of data encryption for storage and network transactions. This enables wide use of encrypted data in transit.
Intel® AVX/AVX2	New 256-bit instruction set extensions for Intel® Streaming SIMD Extensions (Intel® SSE).	Improves the performance of floating-point and integer operations, as well as operations that contain array and string manipulations.
Intel® Transactional Synchronization Extensions (Intel® TSX)	Support for Java synchronization through an Intel TSX JVM option.	Provides the benefit of fine-grain lock performance with coarse-grain locks, when you use the -XX:+UseRTMLocking JVM option.
BMI1 Instructions	The Just-in-time compiler recognizes the patterns for ANDN (X & !Y), BLSR (X & (X - 1)), BLSMSK (X ^ (X - 1)) and BLSI (X & ~X) instructions. The JIT also generates an intrinsic function for Integer.numberOfLeadingZeros using LZCNT instructions. It also generates an intrinsic Integer.numberOfTrailingZeros function using TZCNT instructions.	Improves the performance of bit manipulation code sequences that are used in Java applications.
Fast cyclic redundancy check (CRC) computation	New CRC generation algorithm using the carry-less multiplication instruction PCLMULQDQ.	Increases file checksum and compression/decompression checksum performance, which increases network and file system performance.
ISO 8859-1 encoding	Uses Intel AVX/Intel AVX2 to increase character encoding to 16 or 32 characters at a time, instead of character by character.	Improves the performance of applications that use international string encoding.
BigInteger multiply	Uses 64-bit integer multiply instructions with 128-bit results, in order to perform BigInteger Multiply. This optimization also uses ADCX/ADOX instructions where they are available.	Improves the performance of the BigInteger Multiply operation, which is used in RSA cryptography and multi-precision arithmetic.
1GB large pages	Supports 1GB large memory pages in the JVM runtime for both code and data memory.	Increases the performance of large analytics jobs.

Table 3. Java* development kit (JDK) 7 and earlier optimizations that help boost Java performance

JDK 7 and earlier optimizations	Description	Performance benefit
Instruction decoding optimization	Removes length-changing prefixes and aligning branch targets.	Allows the Java* virtual machine (JVM) to take advantage of new Intel architecture performance features
Out-of-order execution optimization	Eliminates partial flag and partial register stalls.	Allows the JVM to take advantage of new Intel architecture performance features.
Intrinsic function optimization	Optimizes the most common code paths.	Reduces instruction count and cycles per instruction (CPI), which increases overall performance.
Allocation pre-fetch	Caches additional memory ahead of the object being allocated.	Increases performance of critical code paths.
Large-page usage	Uses 2MB large memory pages for both code and data memory.	Increases performance of large analytics jobs.
Vectorization	Loads, operates on, and stores multiple array elements within a single machine instruction.	Increases execution performance of widely used operations, which improves overall performance.
Compressed references	Stores 32-bit compressed references for 64-bit pointers.	Reduces memory footprint and cache misses while improving cycles per instruction (CPI).
Lock optimization	Multi-tiered lock deferrals to avoid or delay inflation to fat locks for coincidental lock contention.	Increases scalability through reduction of fat locks.
HashMap, TreeMap, BigDecimal	Optimizes commonly used functions.	Improves performance by using caching and other optimizations to reduce path length and object allocation.
String object allocation optimizations	Optimizes code generation for methods that are frequently used.	Reduces intermediate object allocation for common operations, such as string comparisons.

DEVELOPER TOOLS ENHANCE JAVA CODE OPTIMIZATION

Along with collaborating with Oracle and OpenJDK on JVM development, Intel contributes across the entire Java ecosystem, including contributions to tools and libraries. This work helps support Java implementations so you can gain the maximum benefit when running your applications on Intel processors.

For information about specific Intel software packages, please visit the Intel Developer Zone.

Intel® VTune™ Amplifier support for Java

Intel® VTune Amplifier helps you optimize your code by providing you with code profiling and analysis tools. These help you collect a rich set of data for your application and analyze both serial and parallel performance. Using Intel VTune Amplifier, you can better tune your application for CPU performance, multi-core scalability, bandwidth, and more. Intel VTune Amplifier also provides tools to sort, filter, and visualize collected data for quick insight into performance bottlenecks in any Java application.

Intel VTune Amplifier now supports both Oracle and OpenJDK. Learn more at http://software.intel.com/en-us/intel-vtune-amplifier-xe/

Intel® Math Kernel Library

Intel® Math Kernel Library (Intel® MKL) includes highly vectorized and threaded linear algebra, fast Fourier transforms (FFT), vector math, and statistics functions. Through a single C or Fortran API call, these functions automatically scale across previous, current, and future processor architectures by selecting the best code path for each architecture. Using Java native interface, a Java application can also bind with Intel MKL and use its rich set of features. The Intel MKL includes a set of Java examples to demonstrate the use of MKL functions from Java.

Intel® MKL-DNN

The Intel® Math Kernel Library for Deep Neural Networks (Intel® MKL-DNN) is an open source performance library for deep learning (DL) applications. It is intended to accelerate DL frameworks on Intel architecture.

The Intel MKL-DNN includes highly vectorized and threaded building blocks to implement convolutional neural networks (CNN). It is highly optimized to accelerate compute-intensive parts of deep learning applications. In particular, it helps accelerate DNN frameworks such as Caffe*, Google Tensorflow*, Theano*, and Torch*.

The library is implemented in C++, and provides both C++ and C APIs, which allow the functionality to be used from a wide range of high-level languages, such as Java or Python Software Foundation Python*.

Intel MKL-DNN is distributed as source code via a github repository, and is licensed under the Apache 2.0 license

For more information about this math library, visit https://software.intel.com/en-us/articles/intel-mkl-dnn-part-1-library-overview-and-installation

Intel® Data Analytics Acceleration Library

Intel® Data Analytics Acceleration Library (Intel® DAAL) is the library of Intel architecture-optimized building blocks that cover all stages of data analytics. This includes data acquisition from a data source, and from preprocessing, transformation, data mining, modeling, validation, and decision making.

To achieve the best performance on a range of Intel processors, Intel DAAL uses optimized algorithms from the Intel® Math Kernel Library and Intel® Integrated Performance Primitives. Intel DAAL provides application programming interfaces for C++, Java, and Python* languages.

Learn more about this library at: https://software.intel.com/en-us/intel-daal

BigDL

BigDL is a distributed deep learning library for Apache Spark*. Using Intel BigDL, you can write deep learning applications, such as Scala* or Python programs, and take advantage of the power of scalable Spark clusters. BigDL can run directly on top of existing Apache Spark or Apache Hadoop clusters. The library provides rich deep learning support, efficient scale out, and high performance.

Learn more about Intel BigDL at https://software.intel.com/en-us/articles/bigdl-distributed-deep-learning-on-apache-spark

Intel® IPP

Intel® Integrated Performance Primitives (Intel® IPP) offer you high-quality, production-ready, low-level software building blocks. These primitives are multi-core, multi-OS, and multi-platform ready. Intel IPP are also royalty-free.

The Intel IPP APIs are specifically designed for certain types of applications. These include image processing, signal processing, and data processing (data compression/decompression and cryptography). Intel IPP are also highly optimized using Intel SSE, Intel AVX, and Intel AVX2 instruction sets, so your application can perform faster than with the code that an optimized compiler can produce by itself.

Basically, Intel IPP is a one-stop shop for programming tools and/or a library that is highly optimized for a wide range of Intel architecture. These APIs have already been used by software developers, integrators, and solution providers. With IPP, you can tune your applications to get the best performance on a given platform, reduce software development and maintenance costs, and help minimize time to market.

Intel IPP also provides optimizations for some open source libraries, such as the well-known Zlib* library (http://zlib.net). For example, the data compression domain of the Intel IPP library contains several functions that can speed up Zlib in both data compression and decompression operations.

Intel provides patch files for the Zlib source to provide drop-in optimization with the Intel IPP functions. The patches support Zlib version 1.2.5.3, 1.2.6.1, 1.2.7.3 and 1.2.8. With JDK 9 support of java.util.zip compression through the system library, you can now use Intel IPP acceleration of compression along with Java, through the LD_PRELOAD and LD_LOAD_LIBRARY mechanism.

To learn more about building a faster Zlib with Intel IPP, visit https://software.intel.com/en-us/articles/how-to-use-zlib-with-intel-ipp-opertimization

Since the 2017 Update 2 release, the Intel IPP libraries can be easily installed via YUM/APT repositories. You can call Intel IPP functions in Java applications through the Java Native Interface (JNI).

SUMMARY

Intel processor-based servers continue to boost the performance of Java and JVM-based applications in the enterprise, thanks in part to the joint efforts of Intel, Oracle, and the Java community. Together, this teamwork in software optimizations has, since 2006, enabled a dramatic 110x performance improvement for Java applications that run on Intel architecture.[1,2] Intel will continue to help optimize the Java software stack for Intel platforms by continuing its strong collaboration with Oracle and the OpenJDK community.

With Java a strong force in all sectors and markets, Intel will also continue to introduce powerful new Java-oriented enhancements and capabilities in each new generation of processors. Currently Intel offers a range of servers based on Intel Xeon processors and Intel Xeon Phi processors that help deliver the efficiencies and performance you need.

By combining the right choice of Intel processor-based server with the latest optimized JVM, enterprise IT depart-ments and cloud service providers can reap significant benefits:

- Improved user-response times with less infrastructure investment
- Improved power/heat manage-ment, leading to lower utility costs
- Improved security capabilities
- Improved scalability

Run your applications with the latest Java release on the newest Intel processors to gain the maximum benefit of today's hardware and software Java optimizations.

For more information about Java and Intel's enabling efforts for Java applications, please visit the **Intel Java Resource Center**

To learn more about Intel's collaborations with other runtime/application communities, visit the **Intel Developer Zone Runtimes**

To see the latest performance for OpenJDK, including a history of the performance characteristics of the nightly upstream Java builds, visit the **Intel Runtime Languages Performance Portal**

1 Based on Standard Performance Evaluation Corporation SPECjEnterprise2005* published results for Oracle Java*. "**SPECjbb2005 (Java Server Benchmark)** is SPEC's benchmark for evaluating the performance of server side Java. Like its predecessor, SPECjbb2000, SPECjbb2005 evaluates the performance of server side Java by emulating a three-tier client/server system (with emphasis on the middle tier)." — https://www.spec.org/jbb2005/

Configurations:
http://www.spec.org/osg/jbb2005/results/res2008q1/jbb2005-20080208-00452.html
http://www.spec.org/osg/jbb2005/results/res2009q2/jbb2005-20090330-00702.html
http://www.spec.org/osg/jbb2005/results/res2010q3/jbb2005-20100629-00874.html
http://www.spec.org/osg/jbb2005/results/res2012q1/jbb2005-20120306-01056.html
https://www.spec.org/jbb2005/results/res2013q4/jbb2005-20130911-01180.html

2 Based on Standard Performance Evaluation Corporation SPECjbb2015* published results for Oracle Java*. "The SPECjbb(r)2015 benchmark has been developed from the ground up to measure performance based on the latest Java application features. It is relevant to all audiences who are interested in Java server performance, including Java virtual machine (JVM) vendors, hardware developers, Java application developers, researchers and members of the academic community." - https://www.spec.org/jbb2015/

Configurations:
https://www.spec.org/jbb2015/results/res2015q3/jbb2015-20150904-00006.html
https://www.spec.org/jbb2015/results/res2016q1/jbb2015-20151214-00037.html
https://www.spec.org/jbb2015/results/res2017q2/jbb2015-20170524-00158.html
https://www.spec.org/jbb2015/results/res2017q3/jbb2015-20170622-00196.html

3 Based on Standard Performance Evaluation Corporation SPECjEnterprise2010* published results for Oracle Java*. "The SPECjEnterprise2010 benchmark is a full system benchmark which allows performance measurement and characterization of Java EE 5.0 servers and supporting infrastructure such as Java virtual machine (JVM), Database, CPU, disk and servers." — http://www.spec.org/jEnterprise2010/

Configurations:
https://www.spec.org/jEnterprise2010/results/res2015q2/jEnterprise2010-20150318-00053.txt
https://www.spec.org/jEnterprise2010/results/res2016q3/jEnterprise2010-20160622-00061.txt
https://www.spec.org/jEnterprise2010/results/res2013q3/jEnterprise2010-20130904-00046.txt
https://www.spec.org/jEnterprise2010/results/res2012q3/jEnterprise2010-20120626-00033.txt
https://www.spec.org/jEnterprise2010/results/res2011q3/jEnterprise2010-20110727-00023.txt

4 Available on select Intel® processors. Requires an Intel® Hyper Threading Technology (Intel® HT Technology)-enabled system. Consult your PC manufacturer. Performance will vary depending on the specific hardware and software used. For more information including details on which processors support Intel HT Technology, visit http://www.intel.com/info/hyperthreading.

5 The 5 percent to 35 percent performance increase seen with Intel® Hyper-Threading Technology is based on internal Intel measurements using a wide variety of Oracle Java* workloads and benchmarks.

6 Requires a system with Intel® Turbo Boost Technology. Intel Turbo Boost Technology and Intel Turbo Boost Technology 2.0 are only available on select Intel® processors. Consult your system manufacturer. Performance varies depending on hardware, software, and system configuration. For more information, visit http://www.intel.com/go/turbo.

7 Up to 50% lower platform idle power based on Intel internal measurement (Feb 2009). Configuration details: Intel internal measurements of 221W at idle with Super Micro Computer, Inc., Supermicro* 2xE5450 (3.0GHz 80W) processors, 8x2GB 667MHz FBDIMMs, 1x700W PSU, 1x320GB SATA hard drive vs. 111W at idle with Supermicro* software development platform with 2xE5540 (Intel® Xeon® processor x5570 2.53GHz 80W) processors, 6x2GB DDR3-1066 RDIMMs, 1x800W PSU, 1x150GB 10k SATA hard drive. Both systems were running Microsoft Windows* 2008 with USB suspend select enabled and maximum power savings mode for PCIe link state power management.

8 Cross-client claim based on lowest performance data number when comparing desktop and mobile benchmarks. Configurations and performance test as follows:

Mobile - Comparing pre-production Intel® Core™ i5-2410M Processor (2C4T, 2.3GHz, 3MB cache), Intel Emerald Lake CRB, 4GB (2x2GB) PC3-10700 (DDR3-1333)-CL9, Hitachi Travelstar* 320GB hard-disk drive, Intel® HD Graphics 3000, Driver: 2185 (BIOS:v.34, Intel v.9.2.0.1009) , Microsoft* Windows* 7 Ultimate 64-bit RTM Intel® Core™2 Duo Processor T7250 (2M Cache, 2.00 GHz, 800 MHz FSB), Intel® Silver Cascade Fab2 CRB, Micron* 4 GB (2x2GB) PC3-8500F (DDR3-1066)-400, Hitachi* 320GB hard-disk drive, Mobile Intel® 4 Series Express Chipset Family w/ 8.15.10.2182 (BIOS: American Megatrends* AMVACRB1.86C.0104.B00.0907270557, 9.1.2.1008).

Desktop - Pre-production Intel® Core™ i5-2400 Processor (4C4T, 3.1GHz, 6MB cache), Intel Los Lunas CRB, Micron* 4GB (2x2GB) PC3-10700 (DDR3-1333)-CL9, Seagate* 1 TB, Intel® HD Graphics 2000, Driver: 2185 (BIOS:v.35, Intel® v.9.2.0.1009), Microsoft Windows 7 Ultimate 64-bit RTM Intel® Core™ 2 Duo E6550 (2C2T, 2.33GHz, 4MB cache), Intel® DG945GCL Motherboard, Micron 2GB (2x1GB) DDR2 667MHz, Seagate* 320 GB hard-disk drive, Intel® GMA 950, Driver: 7.14.10.1329, (BIOS:CL94510J.86A.0034, INF: 9.0.0.1011), Microsoft Windows 7 Ultimate 64-bit RTM.

Business productivity claims based on Business Applications Performance Corporation SYSmark* 2007, which is the latest version of the mainstream office productivity and Internet content creation benchmark tool used to characterize the performance of the business client. SYSmark 2007 preview features user-driven workloads and usage models developed by application experts. Multitasking claims based on Futuremark PCMark Vantage*, a hardware performance benchmark for PCs running Windows 7 or Windows Vista*, includes a collection of various single and multi-threaded CPU, graphics, and HDD test sets with a focus on Windows* application tests. Security workload consists of Si Software Sandra* 2010 -AES256 CPU Cryptographic subtest measures CPU performance while executing AES (Advanced Encryption Standard) encryption and decryption algorithm. For more information go to http://www.intel.com/performance.

Software and workloads used in performance tests may have been optimized for performance only on Intel® microprocessors. Performance tests, such as SYSmark and Business Applications Performance Corporation MobileMark*, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to http://www.intel.com/performance.

Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copyright © 2017 Intel Corporation. All rights reserved

Intel, the Intel logo, Intel Core, Xeon, Xeon Phi, and VTune are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Printed in USA XXXX/XXX/XXX/XX/XX Please Recycle XXXXXX-001US